School of Computing and Information Systems
The University of Melbourne
COMP90049 Knowledge Technologies (Semester 1, 2019)
Workshop exercises: Week 10

1. For the following dataset:

apple	*ibm*	*lemon*	*sun*	CLASS
TRAINING INSTANCES				
Y	N	Y	Y	FRUIT
Y	N	Y	Y	FRUIT
Y	Y	N	N	COMPUTER
Y	Y	Y	Y	COMPUTER
TEST INSTANCES				
Y	N	Y	Y	?
Y	N	Y	N	?

Use the method of Naive Bayes classification, as shown in lectures, to classify the test instances. Revise some of the assumptions that are built into the model.

- The assumption most central to the formulation of the Naive Bayes classifier that we have discussed is the conditional independence assumption, namely, that each attribute is independent to all of the other attributes, given the class under consideration. This assumption (while false) is necessary to make the problem tractable, where finding reliable estimates of the joint distribution of features requires more data than we are likely to have.
- Another important assumption is in the way we find probabilities from the training data. This is most important for the maximum likelihood estimation we perform over the class priors (in contrast, we hedge our bets on the posterior probabilities of the terms using smoothing).
- The fact that these assumptions are demonstrably untrue makes it seem like the classifier should not be effective at predicting the classes of unseen data. However, Naive Bayes is a pretty solid performer! It turns out that the methodology is robust enough to produce decent predictions, despite small (and predictable) discrepancies in the individual probabilities under consideration.
- There are also a number of more minor assumptions:
    - We assume that the distribution of classes in the test set is (roughly) the same as the distribution of classes in the training set, and also that all of the classes in the test data are attested in the training data. This is often phrased as "the training and test instances were sampled from the same underlying distribution." (In fact, this is an assumption of most supervised machine learning algorithms.)

    - We typically require some kind of assumption for our smoothing method, depending on which smoothing method we actually use.
    - We need to make some assumptions about the distribution of continuous attributes (typically Gaussian) if they exist in our data set.

- Anyway, let's classify these test instances:
- Naive Bayes selects a class c from a set of classes C for a test instance $T = \langle t_1, t_2, \ldots, t_n \rangle$ using a set of training instances D according to:

$$c = \text{argmax}_{c_j \in C} P(c_j) P(T \mid c_j) \tag{1}$$

- We will expand *P(T | c_j)* based on our conditional independence assumption (according the attribute values t_i seen in our test instance):

$$P(T \mid c_j) \approx \prod P(t_i \mid c_j) \tag{2}$$

- We can calculate the prior probabilities of the classes straight from the training data, using maximum likelihood estimation. For *FRUIT*, 2 of the 4 instances are *FRUIT*:

$$P(f) = \frac{2}{4} = \frac{1}{2}$$

- We find the conditional probabilities $P(t_i \mid c_j)$ based on maximum likelihood estimation from the data set, smoothing by replacing 0 values with a small, positive, non-zero value ϵ. (In the real world, we would probably use Laplacian smoothing, or some other more serious smoothing method.)
- To do this, we will observe what proportion of the instances for the given class contain the term that we're looking for. For example, for $P(a = Y \mid f)$: of the two *FRUIT* instances, both of them contain *apple*.

$$\begin{aligned}
P(a = \text{Y} \mid f) &= \frac{2}{2} = 1 \\
P(i = \text{Y} \mid f) &= \frac{0}{2} = 0 \\
P(l = \text{Y} \mid f) &= \frac{2}{2} = 1 \\
P(s = \text{Y} \mid f) &= \frac{2}{2} = 1 \\
P(a = \text{Y} \mid c) &= \frac{2}{2} = 1 \\
P(i = \text{Y} \mid c) &= \frac{2}{2} = 1 \\
P(l = \text{Y} \mid c) &= \frac{1}{2} \\
P(s = \text{Y} \mid c) &= \frac{1}{2}
\end{aligned}$$

- When we substitute these values into (2) above, we will replace 0 values with ϵ.
- We will also need the conjugate probabilities, for example P(t = F | c). To find these, we will observe that $P(a \mid b) + P(\bar{a} \mid b) = 1$. So, $P(t = N \mid c) = 1 - P(t = Y \mid c)$.
- Going all the way back to (1), and substituting our simplification from (2), we will now consider the values for *FRUIT* and *COMPUTER* (they aren't really probabilities any more, but it isn't important now).

$$\begin{aligned}
\text{FRUIT} &: P(a=\mathrm{Y} \mid f)P(i=\mathrm{N} \mid f)P(l=\mathrm{Y} \mid f)P(s=\mathrm{Y} \mid f)P(f) \\
&= P(a=\mathrm{Y} \mid f)(1-P(i=\mathrm{Y} \mid f))P(l=\mathrm{Y} \mid f)P(s=\mathrm{Y} \mid f)P(f) \\
&= 1 \times (1-0) \times 1 \times 1 \times \frac{1}{2} \\
&= \frac{1}{2} \\
\text{COMP} &: P(a=\mathrm{Y} \mid c)P(i=\mathrm{N} \mid c)P(l=\mathrm{Y} \mid c)P(s=\mathrm{Y} \mid c)P(c) \\
&= P(a=\mathrm{Y} \mid c)(1-P(i=\mathrm{Y} \mid c))P(l=\mathrm{Y} \mid c)P(s=\mathrm{Y} \mid c)P(c) \\
&= 1 \times (1-1) \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \\
&\approx 1 \times \epsilon \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \\
&\approx \frac{\epsilon}{8}
\end{aligned}$$

- For this test instance, T_1, we choose *FRUIT*, because $\frac{1}{2} > \frac{\epsilon}{8}$ (because ϵ is small).
- For T_2, the calculations are similar, except for the fact that $s = N$:

$$\begin{aligned}
\text{FRUIT} &: P(a=\mathrm{Y} \mid f)P(i=\mathrm{N} \mid f)P(l=\mathrm{Y} \mid f)P(s=\mathrm{N} \mid f)P(f) \\
&= P(a=\mathrm{Y} \mid f)(1-P(i=\mathrm{Y} \mid f))P(l=\mathrm{Y} \mid f)(1-P(s=\mathrm{Y} \mid f))P(f) \\
&= 1 \times (1-0) \times 1 \times (1-1) \times \frac{1}{2} \\
&\approx \frac{\epsilon}{2} \\
\text{COMP} &: P(a=\mathrm{Y} \mid c)P(i=\mathrm{N} \mid c)P(l=\mathrm{Y} \mid c)P(s=\mathrm{N} \mid c)P(c) \\
&= P(a=\mathrm{Y} \mid c)(1-P(i=\mathrm{Y} \mid c))P(l=\mathrm{Y} \mid c)(1-P(s=\mathrm{Y} \mid c))P(c) \\
&= 1 \times (1-1) \times \frac{1}{2} \times (1-\frac{1}{2}) \times \frac{1}{2} \\
&\approx \frac{\epsilon}{8}
\end{aligned}$$

- And again, this is classified as *FRUIT*.

2. A confusion matrix is a summary of the performance of a (supervised) classifier over a set of development ("test") data, by counting the various instances:

		Actual			
		a	b	c	d
Classified	a	10	2	3	1
	b	2	5	3	1
	c	1	3	7	1
	d	3	0	3	5

(a) Calculate the classification **accuracy** of the system. Find the **error rate** for the system.

- In this context, Accuracy is defined as the fraction of correctly identified instances, out of all of the instances. In the case of a confusion matrix, the correct instances are the ones enumerated along the main diagonal (classified as a and actually a etc.):

$$\begin{aligned} \mathit{Accuracy} &= \frac{\#\text{ of correctly identified instances}}{\text{total }\#\text{ of instances}} \\ &= \frac{10+5+7+5}{10+2+3+1+2+5+3+1+1+3+7+1+3+0+3+5} \\ &= \frac{27}{50} = 54\% \end{aligned}$$

- Error rate is just the complement of accuracy:

$$\begin{aligned} \mathit{Error\ Rate} &= \frac{\#\text{ of incorrectly identified instances}}{\text{total }\#\text{ of instances}} \\ &= 1 - \mathit{Accuracy} \\ &= 1 - \frac{27}{50} = 46\% \end{aligned}$$

(b) Calculate the **precision**, **recall**, **F-score** (where β = 1), **sensitivity**, and **specificity** for class d. (Why can't we do this for the whole system? How can we consider the whole system?)

- Precision for a given class is defined as the fraction of correctly identified instances of that class, from the times that class was attempted to be classified. We are interested in the true positives (TP) where we attempted to classify an item as an instance of said class (in this case, d) and it was actually of that class (d): in this case, there are 5 such instances. The false positives (FP) are those items that we attempted to classify as being of class d, but they were actually of some other class: there are 3 + 0 + 3 = 6 of those.

$$\begin{aligned} \mathit{Precision} &= \frac{\text{TP}}{\text{TP}+\text{FP}} \\ &= \frac{5}{5+3+0+3} \\ &= \frac{5}{11} \approx 45\% \end{aligned}$$

- Recall for a given class is defined as the fraction of correctly identified instance of that class, from the times that class actually occurred. This time, we are interested in the true positives, and the false negatives (FN): those items that were actually of class d, but we classified as being of some other class; there are 1 + 1 + 1 = 3 of those.

$$\begin{aligned} \textit{Recall} &= \frac{\text{TP}}{\text{TP}+\text{FN}} \\ &= \frac{5}{5+1+1+1} \\ &= \frac{5}{8} \approx 62\% \end{aligned}$$

- F-score is a measure which attempts to combine Precision (P) and Recall (R) into a single score. In general, it is calculated as:

$$F_\beta = \frac{(1+\beta^2)P \cdot R}{(\beta^2 \cdot P) + R}$$

- By far, the most typical formulation is where the parameter β is set to 1: this means that Precision and Recall are equally important to the score, and that the score is a harmonic mean. • In this case, we have calculated the Precision of class d to be $\frac{5}{11}$ and the Recall to be $\frac{5}{8}$. The F-score where (β = 1) of class d is then:

$$\begin{aligned} F_{\beta=1} &= \frac{2 \cdot P \cdot R}{P+R} \\ &= \frac{2 \cdot \frac{5}{11} \cdot \frac{5}{8}}{\frac{5}{11} + \frac{5}{8}} \\ &= \frac{50}{95} \approx 53\% \end{aligned}$$

- Sensitivity is defined the same way as Recall: $\frac{TP}{TP+FN}$.
- Specificity is Precision with respect to the negative instances:

$$\begin{aligned} \textit{Specificity} &= \frac{\text{TN}}{\text{TN}+\text{FP}} \\ &= \frac{10+2+3+2+5+3+1+3+7}{10+2+3+2+5+3+1+3+7+3+0+3} \\ &= \frac{36}{42} \approx 86\% \end{aligned}$$

3. How is **holdout** evaluation different to **cross-validation** evaluation?

- In a holdout evaluation strategy, we partition our data into a training set and a test set: we build the model on the former, and evaluate on the latter.
- In a cross-validation evaluation strategy, we do the same as above, but a number of times, where each iteration uses one partition of the data as a test set and the rest as a training set (and the partition is different each time). The evaluation metric is typically averaged across the various partitions.